1. SpringBoot @Value 源码解析

@Value的处理器StringValueResolver初始化时机是PropertySourcesPlaceholderConfigurer#postProcessBeanFactory中

PropertyPlaceholderAutoConfiguration，通过 SpringBoot 的自动装配特性创建的。

@Configuration

@AutoConfigureOrder(Ordered.HIGHEST\_PRECEDENCE)

public class PropertyPlaceholderAutoConfiguration {

@Bean

@ConditionalOnMissingBean(search = SearchStrategy.CURRENT)

public static PropertySourcesPlaceholderConfigurer propertySourcesPlaceholderConfigurer() {

return new PropertySourcesPlaceholderConfigurer();

}

}

1. 在Bean创建过程中的属性装配阶段，委托给AutowiredAnnotationBeanPostProcessor处理属性装配
2. AutowiredAnnotationBeanPostProcessor查找Class中包含@Autowired注解和@Value注解的属性和方法
3. 委托给beanFactory#resolveDependency方法解析依赖
4. BeanFactory从容器中根据name和type查询到所有匹配的依赖，选择出最匹配的。
5. applyPropertyValues 将PropertyValues中的数据装配到Bean对象中

详情查看：<https://www.jianshu.com/p/933669270a9f> 补充

<https://www.cnblogs.com/strongmore/p/16229533.html> 主流程

1. @Value可以实现Apollo 实时更新，@Configurationproperties 不能实现，为什么？

@Value 属性热更新原理

apollo利用BeanFactoryPostProcessor的postProcessBeforeInitialization方法，对spring的bean进行遍历，把bean中加了@Value注解的field和method以SpringValue的形式注册到SpringValueRegistry中。

SpringValue包含了更新值需要的信息，例如@Value加在field上，那么Spring就包含了bean对象，加了注解的Field。@Value加在方法上，则会包含加了注解的Method。

热更新逻辑了：

1、配置发生变更，调用AutoUpdateConfigChangeListener负责更新@Value和@ApolloJsonValue注解的方法和属性

2、从SpringValueRegistry中找到需要更新的SpringValue对象

3、利用反射设置SpringValue持有的bean对象的Field或调用method方法

详情查看<https://blog.csdn.net/zidongxiangxi/article/details/109516458>

<https://juejin.cn/post/6871466689238663175>

@ApolloConfigChangeListener+RefreshScope更新@ConfigurationProperties修饰的配置类

建议重启，不建议

Apollo在配置变化后，只能修改Bean的属性，例如我们数据源的属性发生变化，新创建的Connection对象是没问题的，但是连接池中已经创建的Connection对象相关信息是不能动态修改的，所以依然需要重启应用。

确认是由于@ConditionalOnProperty先于EnableApolloConfiguration执行，导致在执行@ConditionalOnProperty时找不到在apollo中的配置，目前还没有想到比较好的解决方案。

String namespaceWithConditionalProperties = "someNamespace";

Config config = ConfigService.getConfig(namespaceWithConditionalProperties);

context.getEnvironment().getPropertySources().addFirst(new ConfigPropertySource(namespaceWithConditionalProperties, config));

1. 为什么不在 long polling 的返回结果中直接返回更新的结果呢？

1、如果推送消息就是有状态了，做不到幂等了，会带来很多问题，有数据的丢失的情况，如果使用只推送变更通知的话，即使丢失了，还是能在下一次变更的时候达到一个一致的状态

2、加载配置接口是幂等的，推送配置的话就做不到了，因为和推送顺序相关（推送消息就是有状态了，做不到幂等了，会带来很多问题）

3、client 的长轮询，是定时轮训的“实时”补充，通过这样的方式

4、总而言之，就是在满足幂等性，实时性的基础上保持设计的简单

1. Spring DeferredResult

Spring WebMVC 中有多种方式可以用来进行异步请求处理，比较常见的是控制器方法返回DeferredResult类型的返回值。DeferredResult适用于处理客户端轮询的场景，可以实现延时响应客户端的效果，有效避免轮询请求过于频繁造成服务器压力，其使用流程如下。注意，Spring WebFlux 本就将建立连接和请求处理分离开，天然异步，所以未支持 DeferredResult 特性

客户端发起请求

服务端接收请求，并将请求挂起，直到以下两种情况才响应客户端

DeferredResult#setResult()方法被调用，请求被主动唤醒，返回结果

指定时间内DeferredResult#setResult()方法未被调用，超时，返回一个DeferredResult#onTimeout()预设的结果

客户端得到响应，处理此次响应结果

源码详情查看：https://blog.csdn.net/weixin\_45505313/article/details/118194918

1. 配置优先级，命令行-D 配置，正常命令行配置，application配置，bootstrap配置，apollo配置优先级

-D >apollo>命令行>application.properties>application.yml>bootstrap

详情查：https://github.com/apolloconfig/apollo/issues/4367

1. 灾备Config eureka配错复盘

原因灾备Config eureka

1. Eureka 三级缓存

常见面试题：<https://www.sunnymaple.cn/2020/03/04/Eureka%E5%B8%B8%E8%A7%81%E9%9D%A2%E8%AF%95%E9%A2%98%E6%80%BB%E7%BB%93/>

Eureka 服务剔除机制补充：https://juejin.cn/post/7020208871809482766

+ 三级缓存  
<https://blog.csdn.net/Saintmm/article/details/122335819>

| **缓存** | **缓存类型** | **所处类** | **概述** |
| --- | --- | --- | --- |
| **registry 一级缓存** | **ConcurrentHashMap** | **AbstractInstanceRegistry** | **实时更新，又名注册表，UI界面从这里获取服务注册信息；** |
| **readWriteCacheMap 二级缓存** | **Guava Cache（LoadingCache）** | **ResponseCacheImpl** | **实时更新，缓存时间180秒；** |
| **readOnlyCacheMap** 三级缓存 | ConcurrentHashMap | ResponseCacheImpl | 周期更新，默认每**30s**从二级缓存readWriteCacheMap中同步数据更新； Eureka Client默认从这里获取服务注册信息，可配为直接从readWriteCacheMap获取 |